



TITLE:

ループプログラムのハードウェア 化(計算アルゴリズムと計算量の基 礎理論)

AUTHOR(S):

阿曾, 弘具

CITATION:

阿曾, 弘具. ループプログラムのハードウェア化(計算アルゴリズムと計算量の基礎理論). 数理解析研究所講究録 1988, 666: 105-114

ISSUE DATE:

1988-07

URL:

<http://hdl.handle.net/2433/100680>

RIGHT:

ループプログラムのハードウェア化

東北大・工 阿曾弘具 (Hiroto Aso)

あらまし 与えられたループプログラムに内在する計算の並列性を抽出し、それから基本計算素子の結線構造を導く方法を与える。このために、並列処理機構における結線構造の表現法を与え、基本的性質を示す。

1. 一様結線構造

並列処理機構は、基本計算素子からなる計算セルをいくつか並べ、通信線で互いに結合したものとみることができる。計算セルはいくつかの入力ポートと出力ポートを持つ。それぞれの集合を P_{in}, P_{out} と記す。出力ポートは複数の入力ポートに結線されるかも知れないが、入力ポートへはその性質上2つ以上の出力ポートをつなぐことはできない。したがって、結線は、入力ポートをどのセルのどの出力ポートにつなぐかということを定める写像で定義できる。

この結線構造は、ラベル付き有向グラフ $\langle V, E, S, \ell \rangle$ によって表現できる。すなわち、 V はセルを置く点の集合、 E はセル間結線を示す有向辺の集合である。有向グラフとして連結であるとする。 S は入力ポートを示すラベルの集合で、 ℓ は各辺がどの入力ポートの結線かを示すための E から S への関数でラベリング関数とよぶ。 $v \in V$ へはいる辺の集合を $\text{Ein}(v)$ とおく。すなわち、

$$\text{Ein}(v) = \{e \mid e = (v', v)\}.$$

各セルの入力ポートはすべて区別できるから、 $\ell: \text{Ein}(v) \rightarrow S$ とみたとき、1対1であるとする。

結線構造の一様性は、グラフのある点から他の点をみたときの関係がどの点においても同一であるということによって定義できるだろう。まず、ある点から他の点を見るという関係を定式化しよう。各ラベル $s \in S$ を次のように定義される写像と同一視する。

$$s(v) = \begin{cases} v' & \text{if } \ell(v', v) = s \\ \perp & \text{if } \ell(v', v) = s \text{ を満たす } v' \text{ が存在しない.} \end{cases}$$

$V_{ex} = V \cup \{\perp\}$ とおくと、 s は V_{ex} から V_{ex} への写像とみなせる ($s(\perp) = \perp$)。以下では、 $s(v)$ を $v \cdot s$ と記す。この記法から、 \cdot を $V \times S$ から V への写像ともみなし、入力近傍写像という。入力近傍写像が定まると、 E が一意に決まる。従って、結線構造は3項組 $\langle V, S, \cdot \rangle$ であると定めてもよい。

ここで、 \perp は入力ポートへの結線が考えている並列処理機構の外部となっていることを意味する。従って、 V は次の2つの集合に分類できる。

$$V_b = \{v \mid \exists s; s(v) = \perp\} = \{v \mid \ell(\text{Ein}(v)) \neq S\},$$

$$V_c = \{v \mid \forall s; s(v) \neq \perp\} = \{v \mid \ell(\text{Ein}(v)) = S\}.$$

V_b を境界節点の集合、 V_c を内部節点の集合という。有限個の点の集合を考えると、一般的には端の点が存在し、端の点と内部の点とでは、結線構造が変わらざるを

得ない。上の2つの集合はこのことの表現となっている。

さて、 Vex 上の写像の集合を考えると、その上に写像の合成による演算が定義できる。 S からその合成演算によって生成される Vex 上の写像の集合は単位的半群 (自由モノイド) となる (0 回の演算の適用を考え、 Vex 上の恒等写像も付加される)。この半群を S^* と記す。 S^* の元は S の元からなる系列によって表現できる。すなわち、 $\sigma = s_1 s_2 \cdots s_n \in S^*$ に対して、

$$v \circ \sigma = (\cdots ((v \circ s_1) \circ s_2) \circ \cdots) \circ s_n.$$

これは意味的には、 v から入力ポート s_1, s_2, \dots, s_n を次々にたどって行っていく先が $v \circ \sigma$ であることを示す。このたどり方 σ と行き着く先の節点 $v \circ \sigma$ に関する構造が v から他の点をみた関係を示すと考えられる。従って、結線が一樣であるとは、2つの $\sigma_1, \sigma_2 \in S^*$ について、ある $v' \in V$ に対して $v' \circ \sigma_1 = v' \circ \sigma_2 \in V$ ならば、任意の $v \in V$ に対して $v \circ \sigma_1 \equiv v \circ \sigma_2$ が成立することであると定義する。但し、 \equiv は両辺の項が共に V の元であれば等しいということを表す。この後半の条件を満たす σ_1, σ_2 について、 $\sigma_1 \equiv \sigma_2$ と記す。 S^* はこの同値関係 \equiv により同値類に分割できる。その同値類の集合 S^\equiv は、 $\{v \circ \sigma \mid \sigma \in S^*\}$ に1対1に対応する。この集合は v における計算処理に影響を与えるセルの配置を与えており、 v への (情報) 流域と呼ぶ。流域の構造を考えることは S^\equiv の構造を考えることに等しい。

【命題1】 $|S| = d$ とする。 S^* が交換律を満たすならば、 S^\equiv は d 次元格子空間に埋め込むことができる。

(証明) $S = \{s_1, s_2, \dots, s_d\}$ とする。 $\sigma \in S^*$ について、 d 次元ベクトル $c(\sigma)$ を次のように定める。第 i 要素 $c(\sigma)_i$ は σ の中の s_i の数である。 σ の交換律による同値類 $[\sigma]$ の各元 σ' に対して、 $c(\sigma') = c(\sigma)$ となり、また、逆も成立する。従って、同値類 $[\sigma]$ と d 次元ベクトル $c(\sigma)$ とは1対1に対応し、本命題が証明された。 \square

Z を整数の集合とする。

例1. 2分木。

$$V = \{1, 2, 3, 4, \dots\}, \quad S = \{R, L\}.$$

$$v \circ R = 2v, \quad v \circ L = 2v+1.$$

S^* は交換律を満たさない。しかし、一樣である。

例2. 2次元メッシュ。

$$V = Z^2, \quad S = \{N, W\}.$$

$$v \circ N = v - (1, 0), \quad v \circ W = v - (0, 1).$$

これは次の交換律をみたす。

$$v \circ NW = v \circ WN (= v - (1, 1)).$$

以上に述べた一般的な結線構造を持つハードウェアを次のように定義する。

【定義1】シストリック処理系 A とは、次のセル機能 F 、セル空間 V 、セル間結線 W 、配置領域 R の4項組 $\langle F, V, W, R \rangle$ である。

(1) $F = \langle Pin, Pout, D, f \rangle$. $Pin, Pout$ は入力・出力ポート集合で、 $|Pin| = n$, $|Pout| = m$ とおく。 D はデータの集合で、 $f = \langle f_1, f_2, \dots, f_a, \dots, f_m \rangle$ は f_a :

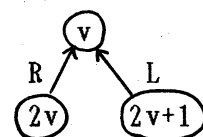


図1. 2分木

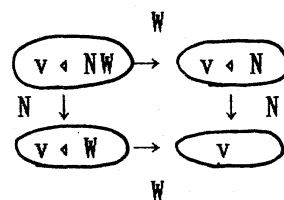


図2. メッシュ

$[P_{in} \rightarrow D] \rightarrow D$ なる写像の組である. $[P_{in} \rightarrow D] = D^n$ は入力ポート上のデータの集合を表し, その元 d を $\langle d(i) \mid i \in P_{in} \rangle$ とも表す. f_a はその入力データに対して出力ポート a の値を決める関数で, セル関数と呼ぶ.

(2) V は, セル機能を配置する空間の座標の集合である. (1次元または2次元格子空間や木なども対象とする).

(3) W は, ポート間結線写像 $u: P_{in} \rightarrow P_{out}$ と入力近傍写像 $\alpha: V \times P_{in} \rightarrow V$ との2項組 $\langle u, \alpha \rangle$ である. 各セル α の入力ポート i をセル $\alpha \alpha i$ の出力ポート $u(i)$ に接続することを意味する.

(4) $R \subseteq V$ は, 実際にセル機能をおくセル座標の集合である.

記憶を明示的には定式化していないが, $\alpha \alpha i = \alpha$ なる入力ポート i , 出力ポート $u(i)$ によって表現できる.

シストリック処理系の各時点での動作はデータ転送, 計算処理の2フェーズからなる. すなわち, 時刻 t でのセル入・出力様相を $y^t: P_{in} \times V \rightarrow D$, $x^t: P_{out} \times V \rightarrow D$ として, 次式で動作が定まる.

転送フェーズ: $y^t(i, \alpha) = x^{t-1}(u(i), \alpha \alpha i)$

計算フェーズ: $x^t(a, \alpha) = f_a(\langle y^t(i, \alpha) \mid i \in P_{in} \rangle)$

但し, $\alpha \alpha i \notin R$ のとき, $x^{t-1}(u(i), \alpha \alpha i)$ は外部からの入力である.

2. ループプログラム

ループプログラムの本体はIF文と割当文とからなるものとする. また, IF文の条件判定や割当文の右辺の評価において副作用は生じないものとする. このとき, IF文に対して等価なif-then-else関数を用いることにより, 本体を, 左辺に現れる配列変数(配列要素指定)には同じものがない割当文だけにすることができる. その結果はつぎのように表現できる.

```
for j1 = b1 to e1
  .....
  for jn = bn to en
    for p = 1 to M
      ap[kp(j1, ..., jn)]
        = fp( $\langle a_{p^q}[k_{p^q}(j_1, \dots, j_n)] \mid q \in \text{Occur} \rangle$ )
```

但し, p は割当文の番号で, その集合を $ID = \{1, \dots, M\}$ とおく. この p に関するループは本体を略記したもので, p 番目の割当文で値が更新される配列名が a_p , その要素を指定する添え字式が $k_p(j_1, \dots, j_n)$, その更新の計算式が f_p で, その q 番目の入力となる配列名が a_{p^q} , その添え字式が $k_{p^q}(j_1, \dots, j_n)$ である. f_p は, (j_1, \dots, j_n) に直接依存してもよい. 単純変数は添え字式が定数である配列とみなす. 例えば, 変数 v は $v[1]$ という配列要素指定とみなす. Occur は入力の出現番号の集合である. ループインデックスの集合を,

$$\text{Index} = \{(j_1, \dots, j_n) \mid b_1 \leq j_1 \leq e_1, \dots, b_n \leq j_n \leq e_n\}$$

とおき, その元を j と記す. Index は実行順序を反映する辞書式順序($>$)をもっており, それは $\text{Index} \times ID$ 上に拡張されているものとする.

以下の議論のために, いくつかの写像を定める. まず, 添え字式 k_p, k_{p^q} は

Indexからデータ配置空間 G_D への写像であるが、 $Z^n (\supseteq \text{Index})$ 上に自然に拡張されているともみなす。更に、 $\text{Arg} = \text{ID} \times \text{Occur}$ 、配列名の集合を Sort とおき、写像 $s: \text{ID} \rightarrow \text{Sort}$, $\arg: \text{Arg} \rightarrow \text{Sort}$ をそれぞれ $s(p) = a_p$, $\arg(p, q) = a_{p^q}$ と定める。

ループ本体の各割当文で参照されている配列変数は、入力としてあらかじめ与えられているか、あるいは、どこかのループインデックスで計算されているはずである。その対応を示すため次の関数を導入する。

【定義2】次の関数 $B: \text{Index} \times \text{Arg} \rightarrow (Z^n \cup \{\perp\}) \times (\text{ID} \cup \{0\})$ を、生成時点関数という。 $B(j, \langle p, q \rangle)$ は、 $\arg(p, q) = s(p')$, $k_{p^q}(j) = k_{p'}(j')$, $(j, p) > (j', p')$ を満たす (j', p') が存在すれば ($j' \notin \text{Index}$ かも知れない)、そのうちで最大の順位にあるものを値とし、存在しなければ、 $(\perp, 0)$ を値とする。特に、値の各々を示すため次の記法を用いる。 $j \cdot \langle p, q \rangle = j'$, $\text{st}(j, \langle p, q \rangle) = p'$ 。ドット・を逆行関数 (retrogression), st を文指定関数と呼ぶ。

すなわち、 $\arg(p, q) = s(p') = a$, $k_{p^q}(j) = k_{p'}(j') = \mu$ の場合、 j における文 p の q 番目の参照 $a[\mu]$ の値が、 j' において文 p' で計算されていることを示す。また、このことは j における f_p の計算より、 $j \cdot \langle p, q \rangle$ における $f_{p'}$ の計算が先に実行されなければならないことを意味する。これを計算の依存性という。

$\text{IndexI} = \{j \cdot \langle p, q \rangle \mid j \in \text{Index}, \langle p, q \rangle \in \text{Arg}\}$,

$\text{IndexA} = \text{Index} \cup \text{IndexI} \cup \{\perp\}$

とおく。 $j \cdot \langle p, q \rangle = \perp$ となるのは、配列名 $\arg(p, q)$ を左辺に持つ文がないか、あっても参照されている添え字に一致するものがないことを意味する。すなわち、その参照はループの外で入力として与えられるものである。 \perp でなくて $j \cdot \langle p, q \rangle \notin \text{Index}$ となるのは、本質的には使われないか、あるいは、初期値として値が与えられるものである。

結線構造 $\langle \text{Index} \cup \text{IndexI}, \text{Arg}, \cdot \rangle$ を計算グラフ (またはデータフローグラフ^[9]) ともいう。

3. ループプログラムのハードウェア化

ループプログラムに対してそれを実行するシストリック処理系を構成する。

ループプログラムの本体の各計算式は、各セルのセル関数に対応させる。このとき、各セル関数の出力値が出力ポートに対応し、引数が入力ポートに対応するので、次のようになる。

$P_{\text{out}} = \text{ID}$, $P_{\text{in}} = \text{Arg} = \text{ID} \times \text{Occur}$.

次に、各ループインデックス j における計算を実行する時刻とセル位置を与える計算時刻関数 T_c 、計算セル位置関数 A_c を考える。

$T_c: \text{IndexA} \rightarrow T$, $A_c: \text{IndexA} \rightarrow V$.

以下では、 V の元として記号 α を用いる。

各 f_p の j における計算は、セル $A_c(j)$ で時刻 $T_c(j)$ に実行される。そのために必要な q 番目の入力 $j \cdot \langle p, q \rangle = j'$ なるインデックスで $\text{st}(j, \langle p, q \rangle) = p'$ 番目の計算式で計算されている。従って、セル $A_c(j)$ の $i = \langle p, q \rangle$ という入力ポートはセル $A_c(j \cdot i)$ の出力ポート $p' = u(i)$ に接続する必要がある。また、その計算は時刻 $T_c(j \cdot i)$ に実行され、時刻 $T_c(j)$ までその計算結果を保存してお

かなければならない。すなわち、入力ポート i から計算素子 p の入力 q までに長さ $Tc(j) - Tc(j \cdot i) - 1$ のシフトレジスタを必要とする。以上の状況の下で、セル機能の一様性、すなわち、どの j に対する計算についても結線および関数機能が同一であるという要請を課すると、まず、ループプログラムは次の条件を満たさなければならない。

(C0) $st(j, i) = u(i)$ は j に依存しない。

更に、 Ac, Tc は次の条件を満たさなければならない。

(C1) $Ac(j \cdot i) = Ac(j) \cdot i$.

(C2) $Tc(j) - Tc(j \cdot i) = \delta(i)$ は j に依存しない。

δ を時間調整写像という。(C1)を一様結線条件、(C2)を同一機能条件という。計算の依存性、および、入出力ポート間のデータ転送には1単位時間を必要とするので、次の必要条件が得られる。転送可能条件と呼ぶ。

(C3) $\delta(i) \geq 0$. 特に、 $j \neq j \cdot i$ ならば、 $\delta(i) > 0$.

次に、セル機能を単純にするために、同時刻、同一セルでは、高々1つのインデックスにおける計算しか実行されないことを要請する。これを最簡機能条件という。次のように記述される。

(C4) $Ac(j) = Ac(j')$ ならば $Tc(j) \neq Tc(j')$.

この対偶は、“計算時刻が一致するならば計算セルは異なる”となる。

以上の条件(C1)~(C4)を満たす Ac, \cdot, Tc, δ が存在すれば、それによりシストリック処理系が定まる。以下にこの δ に対する必要条件を明らかにしていくが、その前に明らかな問題点を解決しておく。それは、 $j \cdot i = 1$ の場合、 $\delta(i)$ は j に依存してしまうということである。この場合が生ずるのは、配列 $arg(i)$ が入力として与えられ、ループ本体では値がセットされない場合である。そこで、ループの計算の意味を変え、ことなく次のように値をセットするようにする。すなわち、 $i = \langle p, q \rangle$, $arg(i) = b$ のとき、 $b[k_{p,q}(j)] = b[k_{p,q}(j)]$ という右辺と左辺が同一の項である割当文をあらかじめ付加する。このループプログラムの等価変換を流れ化変換と呼ぶ。入力データをパイプライン的に流すことに対応する。

次は、(C1), (C2) における $\alpha \cdot i$, $\delta(i)$ の定め方から成立する。

【命題2】(1) $j \cdot i = j$ ならば、 $\alpha \cdot i = \alpha$, $\delta(i) = 0$.

(2) 各 $i, i' \in Arg$ について、ある j に対して、

$j \cdot i = j \cdot i'$ ならば、 $i \equiv i'$, $\delta(i) = \delta(i')$.

Arg を上の(2)に従って、同値類分割し、各同値類を e と表す。すなわち、 $i, i' \in e \Leftrightarrow \exists j; j \cdot i = j \cdot i'$. また、 $j \cdot i = j$ となる i が属する同値類を除いた同値類の集合を E とおく。逆行関数は $IndexA \times E$ 上の関数に自然に拡張する。

条件(C2)は δ, Tc を未知関数とする方程式ともみなすことができる。この方程式を解くために、いくつかの定義をする。まず、 $((j \cdot e_1) \cdot e_2) \cdot e_3$ を $j \cdot e_1 e_2 e_3$ と記す。 E の元からなる系列の集合を E^* と記し、その元を $e_1 e_2 \dots e_n$ あるいは σ と記す。逆行関数は $IndexA \times E^*$ 上の関数に自然に拡張される。関数 $c: E^* \rightarrow [E \rightarrow Z]$ を、

$c(\sigma)(e) = (\sigma \text{ 中の } e \text{ の生起回数})$

と定め、更に、次の記法を導入する。

$$(c(\sigma), \delta) = \Sigma \circ c(\sigma)(e) \cdot \delta(e).$$

【命題3】(1) 任意の $j \in \text{Index}$, 任意の $\sigma \in E^*$ に対して,

$$Tc(j) = Tc(j \cdot \sigma) + (c(\sigma), \delta).$$

(2) $\exists j; j \cdot \sigma = j \cdot \sigma'$ かつ $c(\sigma) \neq c(\sigma')$ となる σ, σ' が存在するならば,

$$(c(\sigma) - c(\sigma'), \delta) = 0.$$

(証明) $Tc(j) - Tc(j \cdot e_1) = \delta(e_1),$

$$Tc(j \cdot e_1) - Tc(j \cdot e_1 e_2) = \delta(e_2), \dots,$$

$$Tc(j \cdot e_1 e_2 \dots e_{n-1}) - Tc(j \cdot e_1 e_2 \dots e_n) = \delta(e_n).$$

$\sigma = e_1 e_2 \dots e_n$ に対する上式より, 辺辺の和をとって,

$$Tc(j) - Tc(j \cdot \sigma) = \Sigma_{p=1}^n \delta(e_p).$$

この右辺は和の交換律より $(c(\sigma), \delta)$ に等しい. 従って, (1) が示された. (2) は (1) より明らかである. \square

この(2)は δ に対する制約条件を与える.

IndexA 上に関係 \leftarrow を次のように定める. $j \leftarrow j' \Leftrightarrow \exists \sigma; j \cdot \sigma = j'$. この関係は, 意味的には, j における計算のために j' における計算結果を必要とすることを示し, 依存関係と呼ぶ. この依存関係の反射的交換的推移的閉包をとると IndexA 上の同値関係となり, \simeq と表す. 異なる同値類の元の間には依存関係が存在しないから, それらの計算は独立に実行可能であり, 各々を実現することにより全体が実現できる. そこで, 一般性を失うことなく, \simeq による同値類は1つであるとする. すなわち, ある $j_0 \in \text{Index}$ が存在して, 任意の j に対して $j \simeq j_0$ であるとする.

【命題4】写像 $v: \text{Index} \rightarrow [E \rightarrow Z]$ が存在して,

$$Tc(j) = Tc(j_0) + (v(j), \delta).$$

(証明) $j \simeq j_0$ より, 例えば, $j \cdot \sigma_1 = j_1, j_2 \cdot \sigma_2 = j_1, j_2 \cdot \sigma_0 = j_0$. このとき, 命題2(1)の式を用いて計算して, 次式を得る.

$$Tc(j) = Tc(j_0) + (c(\sigma_1) - c(\sigma_2) + c(\sigma_0), \delta).$$

そこで, 上のような j_0 に到る E 上の適当な系列を選び, $v(j) = c(\sigma_1) - c(\sigma_2) + c(\sigma_0)$ などと定めればよい. \square

$v(j)$ は, j_0 から j までの計算の依存関係の状況を縮約したものとなっており, j の状況ベクトルという. 具体的な $v(j)$ の値の決定は系列の選び方に依存するが, $(v(j), \delta)$ はその系列の選び方に依存しないことが証明から分かる.

命題4は, $Tc(j)$ が定数 $Tc(j_0)$ と δ の線形結合とから定まることを示している. 従って, 制約条件を満たす δ が存在すれば, Tc は一意に定まることが知られた. Ac については次が成立する.

【命題5】(1) 任意の $j \in \text{Index}$, 任意の $\sigma \in E^*$ に対して,

$$Ac(j \cdot \sigma) = Ac(j) \circ \sigma.$$

(2) $\exists j; j \cdot \sigma = j \cdot \sigma'$ ならば $\sigma \equiv \sigma'$.

(証明) 命題3の証明と同様である. \square

以上の議論から, 次が得られる.

【定理1】ループプログラムがシストリック処理系により実現できるためには, 次の条件(a), (b)を満たさなければならない. $i \in \text{Arg}$ である.

(a) 文指定関数 $st(j, i)$ は j に依存しない.

(b) $\exists j; j \cdot \sigma = j \cdot \sigma'$ かつ $c(\sigma) \neq c(\sigma')$ となるすべての $\sigma, \sigma' \in E^*$ に対して, ラベル関数として, $\sigma \equiv \sigma'$ が成立し, かつ,

$$(c(\sigma) - c(\sigma'), \delta) = 0$$

が成立し, さらに, $\delta > 0$ となる δ が存在する. 但し, $\delta: E \rightarrow T$.

また, (b) が成立するならば, 一様結線, 同一機能, 転送可能 の3条件 (C1), (C2), (C3) が成立する.

(証明) 条件 (C0) および命題 3, 5 から明らかである. \square

この定理より, 条件 (a) を満たすループプログラムについて, 条件 (b) を満たす δ と Ac で更に最簡機能条件を満たすものが得られるならば, シストリック処理系が構成できることが分かる.

計算式 f_0 が直接 j に依存してよいとしたが, そのためには各セルはインデックスの値を知ることができなければならない. そこで, 写像 $Ind: T \times V \rightarrow Index$ を $Ind(t, \alpha) = j \Leftrightarrow Tc(j) = t$ かつ $Ac(j) = \alpha$ と定める. この Ind が存在するとき, インデックス利用可能という. すなわち, 各セルは自身の座標 α と時刻 t とから, そのとき計算しているループインデックス j を知ることができ, j に依存する計算を実行できる. 次が成立する.

【命題 6】 Tc, Ac について, 最簡機能条件を満たすならば, インデックス利用可能である. 逆も成立する.

(証明) $Ind(t, \alpha) \in Tc^{-1}(t) \cap Ac^{-1}(\alpha)$ より明かである. \square

与えられたループプログラムからシストリック処理系を設計する手順を効率化し, また, できる処理系をよりよいものにするためのいくつかの結果を示す.

逆行関数を知ってから Arg を同値類分割して E を求め, δ に関する制約条件式を求めたが, Arg をあらかじめ分類しておくことができる.

【命題 7】 (1) 同じ配列名を持ち, 添え字式が等しい参照変数は, その生成割当文より共に前あるいは共に後にあるならば, 等価である.

(2) 添え字式が同一の参照変数は各々の前後にある生成割当文の左辺の添え字式が同種のものからなっているとき, 等価である. すなわち, $k_{p_1 q_1}(j) = k_{p_2 q_2}(j)$, $\{k_{p'} \mid p' \in s^{-1}(\arg(p_1, q_1)), p' < p_1\} = \{k_{p''} \mid p'' \in s^{-1}(\arg(p_2, q_2)), p'' < p_2\}$ かつ, この等式において $p' \geq p_1, p'' \geq p_2$ とした場合にも等号が成立するならば, $j \cdot \langle p_1, q_1 \rangle = j \cdot \langle p_2, q_2 \rangle$.

(証明) 逆行関数の定義から容易に示される. \square

配列名 b の入力変数が m 個あり, その添え字式を $k^{(r)} (r=1 \sim m)$ とおく. 流れ化変換により, m 個の次の形の割当文が付加されたとする.

$$b[k^{(r)}(j)] = b[k^{(r)}(j)]$$

m 個の b の添え字の集合 $k^{(r)}(Index)$ を考える. これらが排反ならば, 配列 b が m 個に分割されて入力されることを意味し, 問題はない. それらが排反でないとき, 例えば, $\mu \in k^{(1)}(Index) \cap k^{(2)}(Index)$ が存在するとき, $b[\mu]$ は (異なる) インデックスについて 2 カ所で値が設定され, 同一のデータを 2 カ所に流すことを意味する. これは入出力を複雑にするので, 一方を省略することを考える.

[分流変換] 参照変数 $b[k^{(1)}(j)], b[k^{(2)}(j)]$ について, それらが (流

れ化変換をする前には) 入力としてだけ現れるとき, 新しい配列名 b' を導入し, $b[k^{(2)}(j)]$ の参照をすべて $b'[k^{(2)}(j)]$ で置き換え, 次の割当文を付加する.

$$b'[k^{(2)}(j)] = \text{if } k^{(1)}(j) = k^{(2)}(j) \text{ then } b[k^{(1)}(j)] \\ \text{else } b'[k^{(2)}(j)]$$

【命題 8】各 $\mu \in GD$ に対して, $j_0 = \min\{j \mid k^{(2)}(j) = \mu\}$ と定めるとき, $k^{(1)}(j_0) = \mu$ ならば, 分流変換はループプログラムの等価変換である.

(証明) 変換後のプログラムにおいて, インデックス j で $b'[k^{(2)}(j)]$ が参照されたとする. 変換の定義および命題の条件より, その参照はもとのプログラムにおける参照 $b[k^{(2)}(j)]$ に一致する. \square

例 3. 総和: $\text{sum} = \sum_{i=1}^n a_i$.

ループプログラム: for $i=0$ to $n-1$ { $w_i = a_{i+1}$; }
 $w_n = 0$; /* necessary for odd n */
 for $j=1$ to $\lceil \log n \rceil$
 { $\text{imax} = \lceil n/2^j \rceil$;
 for $i=0$ to $\text{imax}-1$ { $w_i = w_{2i} + w_{2i+1}$; }
 $w_{\text{imax}} = 0$; }

等価変換結果: for $j=0$ to $\lceil \log n \rceil$
 { $\text{imax} = \lceil n/2^j \rceil$;
 for $i=0$ to imax
 { /*1*/ $a_{i+1} = a_{i+1}$; /*流れ化変換*/
 /*2*/ $w_i = \text{if } j=0 \text{ then } a_{i+1}$
 else if $i=\text{imax}$ then 0
 else $w_{2i} + w_{2i+1}$; } }

添え字式: $p=1$ (a_{i+1}) について, $k_1(j, i) = i+1$, $k_1^{-1}(j, i) = i+1$. $p=2$ (w_i) について, $k_2(j, i) = i$, $k_2^{-1}(j, i) = i+1$ (a_{i+1}), $k_2^2(j, i) = 2i$ (w_{2i}), $k_2^3(j, i) = 2i+1$ (w_{2i+1}). 入力ポートのラベルを, $\langle 1, 1 \rangle = \langle aa \rangle$, $\langle 2, 1 \rangle = \langle wa \rangle$, $\langle 2, 2 \rangle = \langle w0 \rangle$, $\langle 2, 3 \rangle = \langle w1 \rangle$ とおく.

逆行関数: $(j, i) \cdot \langle w0 \rangle = (j', i')$ については, $k_2^2(j, i) = k_2(j', i')$, $(j, i) > (j', i')$ より $i' = 2i$, $j' = j-1$. 従って,

$$(j, i) \cdot \langle w0 \rangle = (j-1, 2i).$$

同様に, $(j, i) \cdot \langle w1 \rangle = (j-1, 2i+1)$,

$$(j, i) \cdot \langle wa \rangle = (j, i),$$

$$(j, i) \cdot \langle aa \rangle = (j-1, i).$$

計算時刻関数: (第2式 ($p=2$) についてのみ考える)

$$T_c(j, i) - T_c((j, i) \cdot \langle w0 \rangle) = \delta(w0),$$

$$T_c(j, i) - T_c((j, i) \cdot \langle w1 \rangle) = \delta(w1) \quad \text{より,}$$

$$T_c(j, 0) = T_c(0, i) + \sum_{p=0}^{j-1} (i_p \cdot \delta(w1) + (1-i_p) \cdot \delta(w0))$$

但し, $i = (i_{j-1} \cdots i_1 i_0)$ [2進表示].

逆行関数の定まり方から命題 3 (2) の条件を満たす σ , σ' は存在しない. 従って, $\delta(w1)$, $\delta(w0)$ は正でありさえすれば任意に定めてよい. 効率の面から,

$$\delta(w1) = \delta(w0) = 1$$

と定める. このとき, $Tc(j, 0) = Tc(0, i) + j$ となる. 任意の i (j ビット) について, $Tc(0, i)$ は同一の時刻となり, それを t_0 とおくと, $Tc(j, 0) = t_0 + j$. 同様にして一般的に, $Tc(j, i) = t_0 + j$.

計算セル位置関数:

$$Ac(j, i) \cdot \langle w0 \rangle = Ac((j, i) \cdot \langle w0 \rangle) = Ac(j-1, 2i)$$

$$Ac(j, i) \cdot \langle w1 \rangle = Ac((j, i) \cdot \langle w1 \rangle) = Ac(j-1, 2i+1)$$

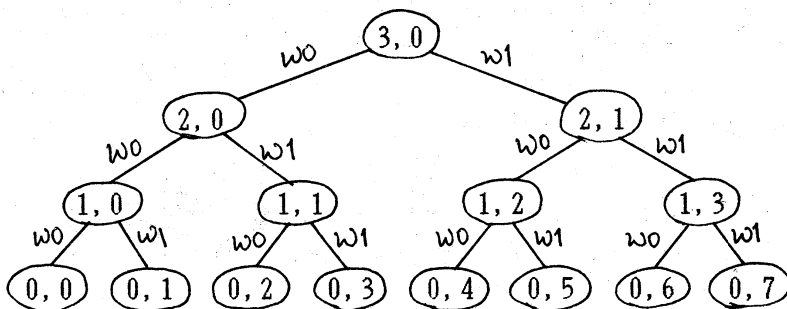
入力近傍関数 \cdot を木のそれにすれば, $Ac(j, i) = i$ は上の式を満たす. 逆に, $Ac(j, i) = i$ とおけば, 上の式から結線構造は木になる (ルートのところが例 1 と若干異なる).

$n=7$ の場合, 計算グラフ, 結線構造は図 3 のようになる.

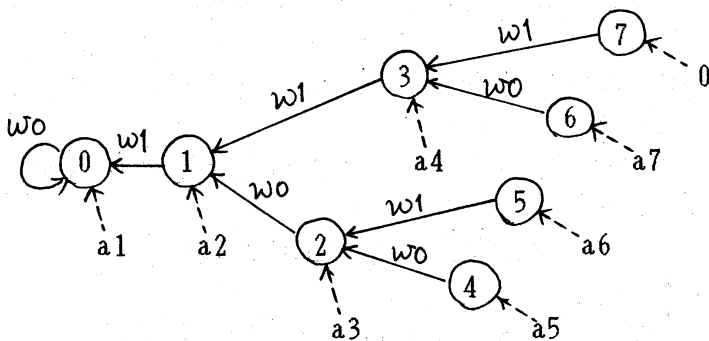
4. むすび

結線構造を定式化し, その一様性について 1 つの考えを示した. それらを基に, シストリックアレーの定義^[10]を若干拡張して, シストリック処理系を定式化した. それは結線構造として線形格子空間だけでなく木などのより一般的なものも対象にしている.

更に, ループプログラムからシストリック処理系を構成する一般的方法を与え, シストリック処理系で実現可能なループプログラムのクラスに関する条件を与え



(a) 計算グラフ (各節点: (j, i))



(b) 結線構造 (各節点: $Ac(j, i) = i$)
(点線: 初期外部入力)

図 3. 総和のシストリック処理系 ($n=7$, $\lceil \log n \rceil = 3$)

た。その条件は、 $j = j \cdot i$ (データ依存ベクトルという) が j に完全に依存することにも許すものとなっており、組織的にハードウェア化できるループプログラムの対象を広げたといえる。従来は、データ依存ベクトルが定数ベクトルの場合にループプログラムや漸化式からシストリックアレーを構成する方法が得られている^[1-9]。また、データ依存ベクトルが j に依存するとしてもその有限的な性質 (ある要素が奇数か偶数かなど) に依存するものであった^[11, 12]。それらは結線構造がアレーであるために課された条件であると考えられ、本論文は結線構造を一般化することによって対象を拡張したものである。

この構成法に並列度や計算時間の評価を導入してより実用的なものにすること、および、入力近傍関数をよりシステムティックに求める方法を見いだすことは、今後の課題である。

謝辞 本研究を進めるうえで御討論頂いた東北大学木村正行教授に感謝する。また、本研究の一部は文部省科学研究費補助金 (総合研究 (A) 課題番号 62302032) の援助を受けた。ここに記して感謝する。

文 献

- [1] D. I. Moldovan; "On the analysis and synthesis of VLSI algorithms", IEEE Trans. Comp. C-31, 11, pp. 1121-1126. (1982)
- [2] D. I. Moldovan; "On the design of algorithm for VLSI systolic arrays", Proc. IEEE, 71, 1, pp. 113-120. (1983)
- [3] 若林, 菊野, 吉田; "漸化式を用いたハードウェアアルゴリズムの設計について", 信学論 D, J67-D, 8, pp. 861-868. (1984)
- [4] P. Quinton; "Automatic synthesis of systolic arrays from uniform recurrent equations", Proc. 11th Int. Symp. Comp. Arch., pp. 208-214. (1984)
- [5] G. J. Li and B. W. Wah; "The design of optimal systolic arrays", IEEE Trans. Comp. C-34, 1, pp. 66-77. (1985)
- [6] D. I. Moldovan, I. A. B. Fortes; "Partitioning and mapping algorithms into fixed size systolic arrays", IEEE Trans. Comp. C-35, 1, pp. 1-12. (1986)
- [7] 三浦, 阿曾, 稲垣; "多重ループプログラムを処理するシストリックアルゴリズムの構成法", 信学論 D, J70-D, 3, pp. 515-524. (1987)
- [8] 堀池, 西田, 坂口; "プロセッサ数の制限下におけるシストリックアレーの設計法", 信学論 D, J70-D, 5, pp. 880-888. (1987)
- [9] 飯国, 酒井, 得丸; "データフローグラフを用いたシストリックアレーの構成法", 信全大 (総合), 1676. (1987)
- [10] 阿曾, 稲垣; "シストリックアルゴリズムの定式化と情報の流れ", 信学論 D, J70-D, 6, pp. 1074-1082 (1987)
- [11] 阿曾; "シストリックアルゴリズム化可能な問題のクラスについて", 夏の LA シンポジウム. (1987)
- [12] 阿曾; "セル構造情報処理系とプログラミング", 昭62電気関係学会東海支部連合大会, S5-1. (1987)

(以上)